Overview of Fortuna Web Application

Developed By Luke Sylvester

Introduction

Fortuna is a web application that is designed to improve the user listening experience by creating Spotify queues that are dynamic and unpredictable. This application uses the Spotify API to integrate with the user's account, allowing them to pick a playlist from their account library, which will then be run through the Fortuna Shuffle Algorithm. This will post a unique queue directly to a user's current session.

The algorithm connects with a weather database that is designed to create random numbers. By using a combination of unpredictable endpoints (ex: wind speed, solar flare intensity, etc), we can create truly random numbers, generated from unpredictable natural occurrences. Each song on a playlist will be assigned a random number using this method and then sorted numerically by a merge sort algorithm. The result will ensure that every song on a playlist will be added to the queue, and that no favoritism is used to curate the queue order.

Required Python Packages

- Flask
- Spotipy
- Requests

Useful Documentation

Flask: https://flask.palletsprojects.com/en/3.0.x/

Spotipy: https://spotipy.readthedocs.io/en/2.22.1/

Spotify API: https://developer.spotify.com/documentation/web-api

Overview of The Flask App Layout

Section 1: Flask Application Initialization and Configuration

Summary: This Section initializes the Flask application instance and configures cookies.

Routes: No specific routes associated.

Section 2: Required Functions

Summary: This Section contains various functions used within the application.

Functions:

`create_spotify_oauth()`: Helps with user authentication/token using Spotipy library.

`get_token()`: Retrieves saved token from cookies.

`get_random_nums(song)`: Retrieves a list of random numbers from random.org API

`merge(left, right)`, `merge_sort(lst)`: Implements merge sort algorithm.

Section 3: Application Routes

Summary: This Section defines routes for different functionalities of the application.

3A: Home Pages

Summary: Routes related to the home pages of the application.

Routes:

/: Home page.

/lets-begin: Transition page for orb animation.

3B: User Authentication

Summary: Routes controlling the login logic of the app, directing users to Spotify's login and authentication pages.

Routes:

/login: Initiates user authentication through Spotify.

/redirect: Handles redirection after user authentication.

3C: Playlist Selection

Summary: Routes for allowing users to select a playlist from their library.

Routes:

/user-playlists: Displays user's playlists.

/open-playlist: Opens a selected playlist.

3D: Randomization

Summary: Route for randomizing tracks in a playlist and adding them to the queue.

Route:

/randomize: Randomizes tracks in a playlist and adds them to the queue.

Overview of Fortuna Web Application

Developed By Luke Sylvester

SECTION 4: Analysis and Experiment Data

Summary: Data and route related to analysis of algorithm performance.

Route:

`/testing`: Gathers the user's current queue and then tally's the songs in the first 10 tracks

Note:

further analysis with the experiment data was done in a separate file named "testing.py".

Testing.py

Experiment Setup:

- The /testing route was used to tally the number of times a song appears in the first 10 tracks of a user's current queue.
- Two algorithms were compared: "Fortuna" and the Spotify algorithm.
- Ten trials were conducted for each algorithm, with playlists of lengths 20, 50, and 100 songs.

Results:

Tally Results:

Variables, spotify_20_songs, fortuna_20_songs, spotify_50_songs, etc., contain the tallies for each trial and playlist depth.

Analysis of Unique Songs:

To assess the diversity of songs introduced into the queue, the code calculated the number of unique songs in each experimental group.

Variables, spotify_num_unique_songs_20, fortuna_num_unique_songs_20, etc., store the count of unique songs for each playlist depth and algorithm combination.

Analysis of High-Frequency Songs:

The code includes loops to determine the number of songs with a tally larger than a threshold of 3 for each experimental group.

Results are stored in variables, spotify_songs_over3_20, fortuna_songs_over3_20, etc.

Overview of Fortuna Web Application

Developed By Luke Sylvester

Visualization:

Matplotlib is used to create bar charts visualizing the experimental results. Two figures were created and saved to the "figures" folder.

Figure 1

Bar chart showing the number of unique songs added to the first 10 tracks of the queue for each experimental group in each condition.

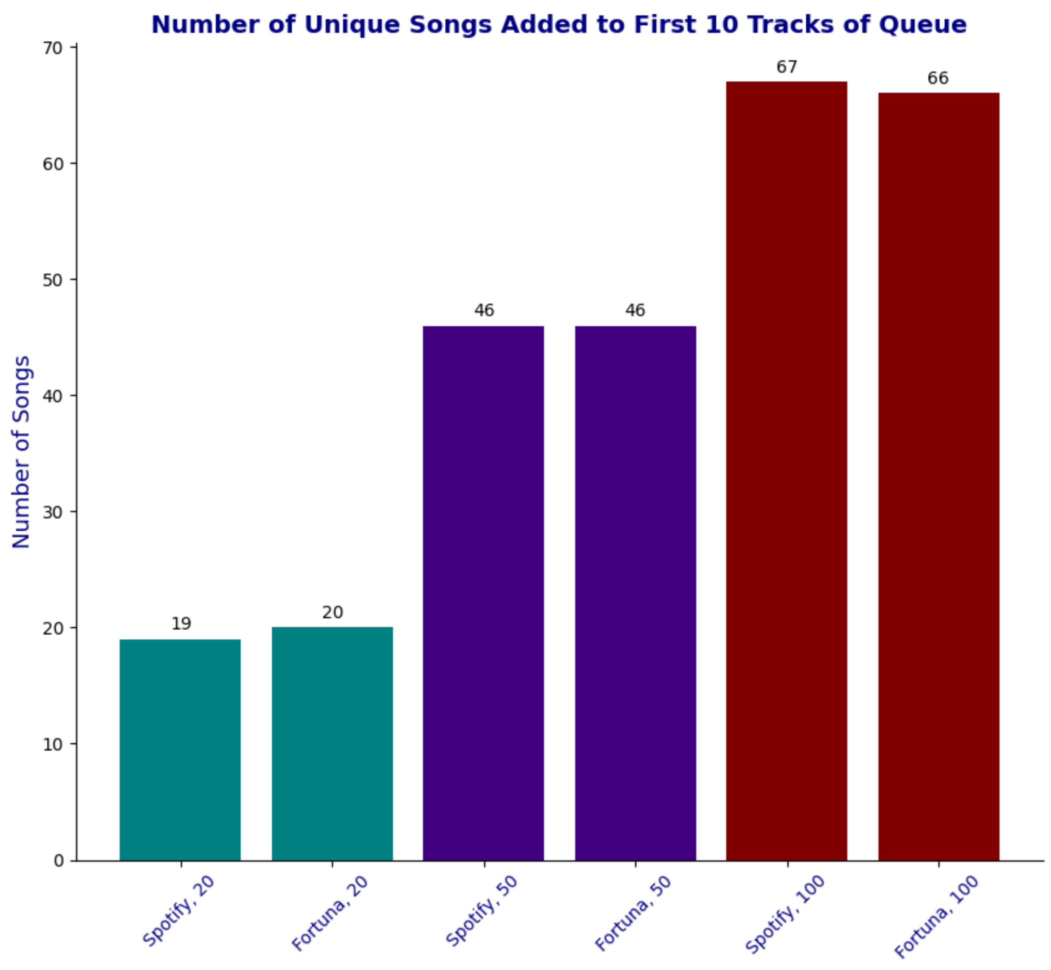
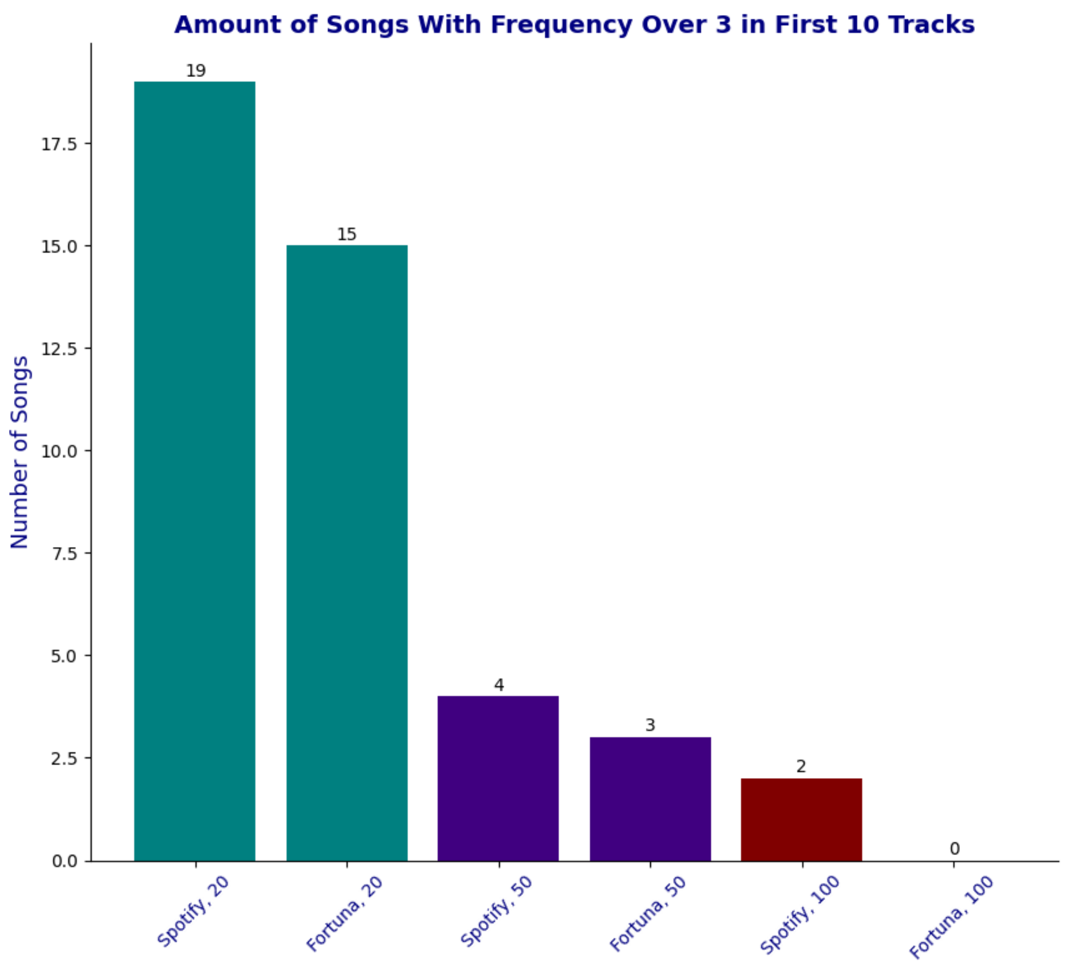


Figure 2

Bar chart showing the number of songs with a frequency over 3 in the first 10 tracks of the queue for each experimental group in each condition.

Overview of Fortuna Web Application

Developed By Luke Sylvester



Static Folder

Static folder holds a single file, “style.css”, which is used to style the website. This file is linked in the head section of the HTML document.

Templates Folder

Templates folder holds 6 html files which each style a different corner of the site. These html templates are rendered by the various flask routes in their “return” statements.